CLAIMS

09826089 . 042501

What is claimed is:

1. A resource queue, comprising:
   (a) a plurality of entries, each entry having unique resources required for information processing;
   (b) the plurality of entries allocated amongst a plurality of independent hardware threads such that the resources of more than one thread may be within the queue; and
   (c) the entries allocated to one thread being capable of being interspersed among the entries allocated to another thread.

2. The resource queue of claim 1, further comprising:
   (a) a first entry of one thread being capable of wrapping around the last entry of the same thread.

3. The queue of claim 1, further comprising:
   (a) a head pointer and a tail pointer for at least one thread wherein the head pointer is the first entry of the at least one thread and the tail pointer is the last entry of the at least one thread, and
   (b) one of the unique resources is a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread.

4. The resource queue of claim 3, further comprising:
   (a) at least one free pointer for the at least one thread indicating an entry in the queue available for resources of the at least one thread.

5. The queue of claim 1, wherein the information processing further comprises:
   (a) an out-of-order computer processor, and
   (b) the resource queue may further comprise a load reorder queue and/or a store reorder queue and/or a global completion table and or a branch information queue.

09825840 .052601

6. A resource queue in an out-of-order multithreaded computer processor, comprising:

(a) a load reorder queue;

(b) a store reorder queue;

(c) a global completion table;

(d) a branch information queue,

at least one of the queues comprising:

(i) a plurality of entries, each entry having unique resources required for information processing;

(ii) the plurality of entries allocated amongst a plurality of independent hardware threads such that the resources of more than one thread may be within the queue; and

(iii) the entries allocated to one thread being capable of being interspersed among the entries allocated to another thread;

(iv) a first entry of one thread being capable of wrapping around the last entry of the same thread;

(v) a head pointer and a tail pointer for at least one thread wherein the head pointer is the first entry of the at least one thread and the tail pointer is the last entry of the at least one thread;

(vi) a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread; and

(vii) at least one free pointer for the at least one thread indicating an entry in the queue available for resources of the at least one thread.

09974925 .100601

7. A method of allocating a shared resource queue for multithreaded electronic data processing, comprising:
   (a) determining if the shared resource queue is empty for a particular thread;
   (b) finding the first entry of a particular thread;
   (c) determining if the first entry and a free entry of the particular thread are the same;
   (d) if, not advancing the first entry to the free entry;
   (e) incrementing a bank number if the first entry passes the last entry before it finds the free entry;
   (f) allocating the next free entry by storing resources for the particular thread.

8. The method of claim 7, further comprising deallocating multithreaded resources in the shared resource queue, comprising:
   (a) locating the last entry in the shared resource queue pertaining to the particular thread;
   (b) determining if the last entry is also the first entry for the particular thread;
   (c) if not, finding the next entry pertaining to the particular thread;
   (d) determining if the bank number of the next entry is the same as the last entry and if so, deallocating the next entry by marking the resources as invalid; and
   (e) if not, then skipping over the next entry and decrementing the bank number;
   (f) finding the next previous entry pertaining to the particular thread.

10852506.042601

9. The method of claim 7, further comprising flushing the shared resource queue, comprising the steps of:
   (a) setting a flush point indicative of an oldest entry to be deallocated pertaining to the particular thread; and
   (b) invalidating all entries between the head pointer and the flush point which have the same and greater bank number than the bank number of the flush point.

10. A shared resource mechanism in a hardware multithreaded pipeline processor, said pipeline processor simultaneously processing a plurality of threads, said shared resource mechanism comprising:
   (a) a dispatch stage of said pipeline processor;
   (b) at least one shared resource queue connected to the dispatch stage;
   (c) dispatch control logic connected to the dispatch stage and to the at least one shared resource queue; and
   (d) an issue queue of said pipeline processor connected to said dispatch stage and to the at least one shared resource queue;

wherein the at least one shared resource queue allocates and deallocates resources for at least two of said plurality of threads passing into said issue queues in response to the dispatch control logic.

09845026.060801

11. An apparatus to enhance processor efficiency, comprising:
   (a) means to fetch instructions from a plurality of threads into a hardware multithreaded pipeline processor;
   (b) means to distinguish said instructions into one of a plurality of threads;
   (c) means to decode said instructions;

(d) means to allocate a plurality of entries in at least one shared resource between at least two of the plurality of threads;

(e) means to determine if said instructions have sufficient private resources and at least one shared resource queue for dispatching said instructions;

(f) means to dispatch said instructions;

(g) means to deallocate said entries in said at least one shared resource when one of said at least two threads are dispatched;

(h) means to execute said instructions and said resources for the one of said at least two threads.

12. The apparatus of claim 11, further comprising:

(a) means to flush the at least one shared resource of all of said entries pertaining to the one of said at least two threads.

13. A computer processing system, comprising:

(a) a central processing unit;

(b) a semiconductor memory unit attached to said central processing unit;

(c) at least one memory drive capable of having removable memory;

(d) a keyboard/pointing device controller attached to said central processing unit for attachment to a keyboard and/or a pointing device for a user to interact with said computer processing system;

(e) a plurality of adapters connected to said central processing unit to connect to at least one input/output device for purposes of communicating with other computers, networks, peripheral devices, and display devices;

(f) a hardware multithreading pipelined processor within said central processing unit to process at least two independent threads of execution, said pipelined processor comprising a fetch stage, a decode stage, and a dispatch stage; and

(g) at least one shared resource queue within said central processing unit, said shared resource queue having a plurality of entries pertaining to more than one thread in which entries pertaining to different threads are interspersed among each other.

14. The computer processor of claim 13 wherein a first entry of one thread may be located after a last entry of said one thread.

15. The computer processor of claim 14, wherein the hardware multithreaded pipelined processor in the central processing unit is an out-of-order processor.

09574609 . 052801